```
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                AAAAAAAAA     TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                AAAAAAAAA     TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                AAAAAAAAA     TTTTTTTTTTTTTTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM  MMM MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM  MMM MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM  MMM MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEEEEEEEEEEE     MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEEEEEEEEEEE     MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEEEEEEEEEEE     MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAAAAAAAAAAAAAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAAAAAAAAAAAAAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAAAAAAAAAAAAAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEEEEEEEEEEEEEE  MMM      MMM  MMM UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA       TTT
EEEEEEEEEEEEEEE  MMM      MMM  MMM UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA       TTT
EEEEEEEEEEEEEEE  MMM      MMM  MMM UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA       TTT
```

```
VV      VV    AAAAAA    XX      XX    CCCCCCCC    OOOOOO    NN      NN  VV      VV  RRRRRRRR    TTTTTTTTTT
VV      VV    AAAAAA    XX      XX    CCCCCCCC    OOOOOO    NN      NN  VV      VV  RRRRRRRR    TTTTTTTTTT
VV      VV  AA      AA  XX      XX  CC          OO      OO  NN      NN  VV      VV  RR      RR      TT
VV      VV  AA      AA  XX      XX  CC          OO      OO  NN      NN  VV      VV  RR      RR      TT
VV      VV  AA      AA    XX  XX    CC          OO      OO  NNNN    NN  VV      VV  RR      RR      TT
VV      VV  AA      AA    XX  XX    CC          OO      OO  NNNN    NN  VV      VV  RR      RR      TT
VV      VV  AA      AA      XX      CC          OO      OO  NN  NN  NN  VV      VV  RRRRRRRR      TT
VV      VV  AA      AA      XX      CC          OO      OO  NN  NN  NN  VV      VV  RRRRRRRR      TT
VV      VV  AAAAAAAAAA    XX  XX    CC          OO      OO  NN    NNNN  VV      VV  RR  RR        TT
VV      VV  AAAAAAAAAA    XX  XX    CC          OO      OO  NN    NNNN  VV      VV  RR  RR        TT
  VV  VV    AA      AA  XX      XX  CC          OO      OO  NN      NN    VV  VV    RR    RR      TT      ....
  VV  VV    AA      AA  XX      XX  CC          OO      OO  NN      NN    VV  VV    RR    RR      TT      ....
    VV      AA      AA  XX      XX    CCCCCCCC    OOOOOO    NN      NN      VV      RR      RR    TT      ....
    VV      AA      AA  XX      XX    CCCCCCCC    OOOOOO    NN      NN      VV      RR      RR    TT      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  VAX$DECIMAL_CONVERT - VAX-11 Packed Decimal Instruction Emulator
0000     2          .IDENT  /V04-000/
0000     3
0000     4  ;
0000     5  ;****************************************************************************
0000     6  ;*                                                                          *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                    *
0000    10  ;*                                                                          *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16  ;*  TRANSFERRED.                                                            *
0000    17  ;*                                                                          *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20  ;*  CORPORATION.                                                            *
0000    21  ;*                                                                          *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    24  ;*                                                                          *
0000    25  ;*                                                                          *
0000    26  ;****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ; Facility:
0000    31  ;
0000    32  ;       VAX-11 Instruction Emulator
0000    33  ;
0000    34  ; Abstract:
0000    35  ;
0000    36  ;       The routines in this module emulate the VAX-11 instructions that
0000    37  ;       convert between packed decimal strings and the various forms of
0000    38  ;       numeric string. These procedures can be a part of an emulator package
0000    39  ;       or can be called directly after the input parameters have been loaded
0000    40  ;       into the architectural registers.
0000    41  ;
0000    42  ;       The input parameters to these routines are the registers that
0000    43  ;       contain the intermediate instruction state.
0000    44  ;
0000    45  ; Environment:
0000    46  ;
0000    47  ;       These routines run at any access mode, at any IPL, and are AST
0000    48  ;       reentrant.
0000    49  ;
0000    50  ; Author:
0000    51  ;
0000    52  ;       Lawrence J. Kenah
0000    53  ;
0000    54  ; Creation Date
0000    55  ;
0000    56  ;       19 October 1983
0000    57  ;
```

```
0000     58 ; Modified by:
0000     59 ;
0000     60 ;       V01-003 LJK0040         Lawrence J. Kenah       24-Jul-1984
0000     61 ;               Longword context instructions (INCL and DECL) cannot be used
0000     62 ;               to modify the sign byte in the destination string for CVTSP.
0000     63 ;
0000     64 ;       V01-002 LJK0024         Lawrence J. Kenah       20-Feb-1984
0000     65 ;               Add code that handles access violations. Perform minor cleanup.
0000     66 ;
0000     67 ;       V01-001 LJK0008         Lawrence J. Kenah       19-Oct-1983
0000     68 ;               The emulation code for CVTPS, CVTPT, CVTSP, and CVTTP
0000     69 ;               was moved into a separate module.
0000     70 ;
0000     71 ;--
```

```
                        .SUBTITLE       Declarations

                ; Include files:

                        .NOCROSS                        ; No cross reference for these
                        .ENABLE         SUPPRESSION     ; No symbol table entries either

                        CVTPS_DEF                       ; Bit fields in CVTPS registers
                        CVTPT_DEF                       ; Bit fields in CVTPT registers
                        CVTSP_DEF                       ; Bit fields in CVTSP registers
                        CVTTP_DEF                       ; Bit fields in CVTTP registers

                        $PSLDEF                         ; Define bit fields in PSL

                        .DISABLE        SUPPRESSION     ; Turn on symbol table again
                        .CROSS                          ; Cross reference is OK now

                ; External declarations:

                        .DISABLE        GLOBAL

                        .EXTERNAL -
                                        VAX$DECIMAL_EXIT,-
                                        VAX$DECIMAL_ACCVIO,-
                                        VAX$ROPRAND

                ; PSECT Declarations:

                        .DEFAULT        DISPLACEMENT , WORD

                        .PSECT _VAX$CODE PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, LONG

                        BEGIN_MARK_POINT
```

```
                                N 15
0000    107
0000    108                 .SUBTITLE       VAX$CVTPx - Convert Packed to Numeric String
0000    109 ;+
0000    110 ; Functional Description:
0000    111 ;
0000    112 ;       The conversion from a packed decimal string to a numeric string
0000    113 ;       (CVTPS and CVTPT instructions) consists of much common code and
0000    114 ;       two small pieces of code that are instruction specific, the
0000    115 ;       beginning and a portion of the end processing. The actual routine
0000    116 ;       exit path is the common exit path from the decimal instruction
0000    117 ;       emulator, VAX$DECIMAL_EXIT.
0000    118 ;
0000    119 ;       The two routines perform instruction-specific operations on the
0000    120 ;       first byte in the stream. The bulk of the work is done by a common
0000    121 ;       subroutine. Some instruction-specific end processing is done before
0000    122 ;       final control is passed to VAX$DECIMAL_EXIT.
0000    123 ;
0000    124 ;       The structure is something like the following.
0000    125 ;
0000    126 ;               CVTPS                                   CVTPT
0000    127 ;               -----                                   -----
0000    128 ;
0000    129 ;       Store sign character                    Store table address
0000    130 ;                                               Unpack registers
0000    131 ;                       !                               !
0000    132 ;                       !                               !
0000    133 ;                       \                               /
0000    134 ;                        \                             /
0000    135 ;                         \                           /
0000    136 ;                          !                         !
0000    137 ;                          v                         v
0000    138 ;
0000    139 ;                   Handle unequal srclen and dstlen
0000    140 ;                   Move all digits except last digit
0000    141 ;
0000    142 ;                       !                               !
0000    143 ;                       !                               \
0000    144 ;                       /                                \
0000    145 ;                      /                                  \
0000    146 ;                     !                                    !
0000    147 ;                     v                                    v
0000    148 ;
0000    149 ;       Move last digit to output               Use table to move last digit
0000    150 ;                                               and sign to output string
0000    151 ;                       !                               !
0000    152 ;                       !                               !
0000    153 ;                       \                               /
0000    154 ;                        \                             /
0000    155 ;                         \                           /
0000    156 ;                          !                         !
0000    157 ;                          v                         v
0000    158 ;
0000    159 ;                           VAX$DECIMAL_EXIT
0000    160 ;                   Set condition codes and registers
0000    161 ;                   to their final values
0000    162 ;
0000    163 ;
```

```
                        0000   164 ;
                        0000   165 ; Input Parameters:
                        0000   166 ;
                        0000   167 ;       See instruction-specific entry points
                        0000   168 ;
                        0000   169 ; Output Parameters:
                        0000   170 ;
                        0000   171 ;       R0 = 0
                        0000   172 ;       R1 = Address of byte containing most significant digit of
                        0000   173 ;            the source string
                        0000   174 ;       R2 = 0
                        0000   175 ;       R3 = Address of lowest addressed byte of destination string
                        0000   176 ;            (See instruction-specific header for details)
                        0000   177 ;
                        0000   178 ; Condition Codes:
                        0000   179 ;
                        0000   180 ;       N <- source string LSS 0
                        0000   181 ;       Z <- source string EQL 0
                        0000   182 ;       V <- decimal overflow
                        0000   183 ;       C <- 0
                        0000   184 ;-
                        0000   185
                        0000   186         .SUBTITLE       Data Declarations / Packed to Numeric String
                        0000   187
                        0000   188 ;+
                        0000   189 ; The following table makes a correspondence between the sixteen possible
                        0000   190 ; packed decimal "digits" and their ASCII representation. It is used to
                        0000   191 ; generate the sign character for leading separate numeric strings and
                        0000   192 ; to generate all character output for the CVTPx common code.
                        0000   193 ;-
                        0000   194
                        0000   195 CVTPx_TABLE:
2D 2B 39 38 37 36 35 34 33 32 31 30  0000   196         .ASCII          /0123456789+-+-++/
                     2B 2B 2D 2B     000C
```

```
                     0010   198          .SUBTITLE       VAX$CVTPS - Convert Packed to Leading Separate Numeric
                     0010   199 ;+
                     0010   200 ; Functional Description:
                     0010   201 ;
                     0010   202 ;       The source packed decimal string specified  by  the  source  length  and
                     0010   203 ;       source  address  operands  is  converted  to  a leading separate numeric
                     0010   204 ;       string.  The destination string specified by the destination length  and
                     0010   205 ;       destination address operands is replaced by the result.
                     0010   206 ;
                     0010   207 ;       Conversion is effected by replacing the lowest  addressed  byte  of  the
                     0010   208 ;       destination  string  with  the ASCII character '+' or '-', determined by
                     0010   209 ;       the sign of the source string.  The remaining bytes of  the  destination
                     0010   210 ;       string  are  replaced  by the ASCII representations of the values of the
                     0010   211 ;       corresponding packed decimal digits of the source string.
                     0010   212 ;
                     0010   213 ; Input Parameters:
                     0010   214 ;
                     0010   215 ;       R0 = srclen.rw          Length in digits of input decimal string
                     0010   216 ;       R1 = srcaddr.ab         Address of input packed decimal string
                     0010   217 ;       R2 = dstlen.rw          Number of digits in destination character string
                     0010   218 ;       R3 = dstaddr.ab         Address of destination character string
                     0010   219 ;
                     0010   220 ; Output Parameters:
                     0010   221 ;
                     0010   222 ;       R0 = 0
                     0010   223 ;       R1 = Address of byte containing most significant digit of
                     0010   224 ;            the source string
                     0010   225 ;       R2 = 0
                     0010   226 ;       R3 = Address of the sign byte of the destination string
                     0010   227 ;
                     0010   228 ; Condition Codes:
                     0010   229 ;
                     0010   230 ;       N <- source string LSS 0
                     0010   231 ;       Z <- source string EQL 0
                     0010   232 ;       V <- decimal overflow
                     0010   233 ;       C <- 0
                     0010   234 ;
                     0010   235 ; Notes:
                     0010   236 ;
                     0010   237 ;       Note that the two entry points VAX$CVTPS and VAX$CVTPT must save the
                     0010   238 ;       exact same set of registers because the two routines use a common exit
                     0010   239 ;       path that includes a POPR instruction that restores registers. In
                     0010   240 ;       fact, by saving all registers, even if one or two of them are not
                     0010   241 ;       needed, we can use the common exit path from this module.
                     0010   242 ;-
                     0010   243
                     0010   244 VAX$CVTPS::
            0FFF 8F  BB  0010   245          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>      ; Save the lot
                     0014   246          ESTABLISH_HANDLER       -               ; Store address of access
                     0014   247                  CONVERT_ACCVIO                  ;  violation handler
 58  50  04  01  EF  0019   248          EXTZV   #1,#4,R0,R8                     ; R8 is byte offset to sign "digit"
                     001E   249          MARK_POINT      CVTPS_ACCVIO
 58  6148   F0 8F  8B  001E   250          BICB3   #^B11110000,(R1)[R8],R8 ; R8 now contains sign "digit"
                     0024   251          MARK_POINT      CVTPS_ACCVIO
      83  D8 AF48  90  0024   252          MOVB    CVTPx_TABLE[R8],(R3)+   ; Store sign character in output string
            008A   30  0029   253          BSBW    CVTPx_COMMON            ; Execute bulk as common code
                     002C   254
```

```
                                002C    255 ;+
                                002C    256 ; The common code routine returns here with the following relevant input.
                                002C    257 ;
                                002C    258 ;       R0      Number of digits remaining in source and destination strings
                                002C    259 ;       R1      Address of least significant digit and sign of input string
                                002C    260 ;       R3      Address of last byte in destination to be loaded
                                002C    261 ;       R8      Sign "digit" from input string
                                002C    262 ;       R11     Saved PSW with condition codes to date (N=0,Z,V,C=0)
                                002C    263 ;
                                002C    264 ;       CVTPS_A_DSTADDR(SP)     Saved R3 at input, address of sign character
                                002C    265 ;
                                002C    266 ;       R4 is a scratch register
                                002C    267 ;
                                002C    268 ; If the input string was negative zero, the sign of the output string must
                                002C    269 ; be changed from "-" to "+". In addition, a check is required to insure that
                                002C    270 ; the Z-bit has its correct setting if this digit is the first nonzero digit
                                002C    271 ; encountered in the input string.
                                002C    272 ;-
                                002C    273
                     50   D5    002C    274         TSTL    R0                      ; Check for no remaining input
                     0F   13    002E    275         BEQL    20$                     ; Skip storing digit if nothing left
                                0030    276         MARK_POINT      CVTPS_ACCVIO
          54   61    04   04    EF  0030    277         EXTZV   #4,#4,(R1),R4           ; Get least significant digit
                     03   13    0035    278         BEQL    10$                     ; Skip clearing Z-bit if zero
                  5B 04   8A    0037    279         BICB    #PSL$M_Z,R11            ; Clear saved Z-bit
                                003A    280
                                003A    281         MARK_POINT      CVTPS_ACCVIO
                63   C2 AF44   90  003A    282 10$:    MOVB    CVTPx_TABLE[R4],(R3)    ; Store final output digit
                                003F    283
                                003F    284 20$:    CASE    R8,LIMIT=#10,TYPE=B,<-  ; Dispatch on sign
                                003F    285                 40$,-                   ; 10 => +
                                003F    286                 30$,-                   ; 11 => -
                                003F    287                 40$,-                   ; 12 => +
                                003F    288                 30$,-                   ; 13 => -
                                003F    289                 40$,-                   ; 14 => +
                                003F    290                 40$,-                   ; 15 => +
                                003F    291                 >
                                004F    292
                   5B   08   88 004F    293 30$:    BISB    #PSL$M_N,R11            ; Set N-bit because sign is "-"
             0B    5B   02   E1 0052    294         BBC     #PSL$V_Z,R11,40$        ; Skip if N-bit set but Z-bit clear
                   5B   08   8A 0056    295         BICB    #PSL$M_N,R11            ; Turn off N-bit if negative zero
             04    5B   01   E0 0059    296         BBS     #PSL$V_V,R11,40$        ; Leave sign alone if overflow occurred
                                005D    297         MARK_POINT      CVTPS_ACCVIO
             0C BE    2B   90  005D    298         MOVB    #^A"+",@CVTPS_A_DSTADDR(SP)     ; Make output sign "+"
                   FF9C'  31  0061    299 40$:    BRW     VAX$DECIMAL_EXIT        ; Exit through common code
```

```
0064    301          .SUBTITLE       VAX$CVTPT - Convert Packed to Trailing Numeric
0064    302 ;+
0064    303 ; Functional Description:
0064    304 ;
0064    305 ;       The source packed decimal string specified  by  the  source  length  and
0064    306 ;       source  address operands is converted to a trailing numeric string.  The
0064    307 ;       destination string specified by the destination length  and  destination
0064    308 ;       address  operands is replaced by the result.  The condition code N and Z
0064    309 ;       bits are affected by the value of the source packed decimal string.
0064    310 ;
0064    311 ;       Conversion is effected by using the highest addressed byte (even if  the
0064    312 ;       source  string  value  is  -0)  of  the  source  string  (i.e., the byte
0064    313 ;       containing the sign and the least  significant  digit)  as  an  unsigned
0064    314 ;       index  into  a 256 byte table whose zeroth entry address is specified by
0064    315 ;       the table address operand.  The byte read out of the table replaces  the
0064    316 ;       least  significant  byte of the destination string.  The remaining bytes
0064    317 ;       of the destination string are replaced by the ASCII  representations  of
0064    318 ;       the  values  of  the  corresponding  packed decimal digits of the source
0064    319 ;       string.
0064    320 ;
0064    321 ; Input Parameters:
0064    322 ;
0064    323 ;       R0 <15:0>  = srclen.rw  Length in digits of input decimal string
0064    324 ;       R0 <31:16> = dstlen.rw  Number of digits in destination character string
0064    325 ;       R1         = srcaddr.ab Address of input packed decimal string
0064    326 ;       R2         = tbladdr.ab Address of 256-byte table used for sign conversion
0064    327 ;       R3         = dstaddr.ab Address of destination character string
0064    328 ;
0064    329 ; Output Parameters:
0064    330 ;
0064    331 ;       R0 = 0
0064    332 ;       R3 = Address of byte containing most significant digit of
0064    333 ;            the source string
0064    334 ;       R2 = 0
0064    335 ;       R3 = Address of most significant digit of the destination string
0064    336 ;
0064    337 ; Condition Codes:
0064    338 ;
0064    339 ;       N <- source string LSS 0
0064    340 ;       Z <- source string EQL 0
0064    341 ;       V <- decimal overflow
0064    342 ;       C <- 0
0064    343 ;
0064    344 ; Notes:
0064    345 ;
0064    346 ;   1.  Note that the two entry points VAX$CVTPS and VAX$CVTPT must save the
0064    347 ;       exact same set of registers because the two routines use a common exit
0064    348 ;       path that includes a POPR instruction that restores registers. In
0064    349 ;       fact, by saving all registers, even if one or two of them are not
0064    350 ;       needed, we can use the common exit path from this module.
0064    351 ;
0064    352 ;   2.  This routine and VAX$CVTTP must have a separate JSB entry point.
0064    353 ;       (Several other routines could use one but it is not required.) Code
0064    354 ;       that uses the emulator through its JSB entry points cannot be
0064    355 ;       redirected to a different entry point when the instruction is
0064    356 ;       restarted after an access violation. The only way that a restart can
0064    357 ;       be distinguished from a first pass is through an internal FPD bit. The
```

```
                                0064   358 ;       original sizes for the five operands for CVTPT and CVTTP require all
                                0064   359 ;       the bits in the four general registers.
                                0064   360 ;
                                0064   361 ;       The FPD bit is stored in bit<15> of the "srclen" operand. In order to
                                0064   362 ;       insure that instructions that enter the emulator through the
                                0064   363 ;       VAX$_OPCDEC exception, rather than through its JSB entry points,
                                0064   364 ;       correctly generate reserved operands for lengths in the range 32768 to
                                0064   365 ;       65535, the internal FPD bit cannot be tested at the VAX$ entry point.
                                0064   366 ;       Thus, the extra entry point is required.
                                0064   367 ;
                                0064   368 ;       Note that this implementation has the peculiar effect that a reserved
                                0064   369 ;       operand exception will not be generated if R0<15:0> contains a number
                                0064   370 ;       in the range 32768 and 32768+31 inclusive.
                                0064   371 ;
                                0064   372 ;   3.  The RESTART entry point is needed because information is saved in
                                0064   373 ;       R0<31:24> if the instruction is interrupted by an access violation.
                                0064   374 ;       This information must be cleared out before the length checks are made
                                0064   375 ;       or a spurious reserved operand exception would result.
                                0064   376 ;-
                                0064   377
                                0064   378 VAX$CVTPT_JSB::
          07 50    0F    E5     0064   379         BBCC    #CVTPT_V_FPD,R0,VAX$CVTPT       ; Have we been here before?
                                0068   380
                                0068   381         ASSUME  CVTPT_B_DELTA_PC EQ 3   ; Make sure that we clear the right byte
                                0068   382
                                0068   383 VAX$CVTPT_RESTART::
     50    FF000000 8F    CA     0068   384         BICL2   #^XFF000000,R0           ; Eliminate delta-PC from "dstlen"
                                006F   385
                                006F   386 VAX$CVTPT::
          0FFF 8F    BB         006F   387         PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>      ; Save the lot
                                0073   388         ESTABLISH_HANDLER       -       ; Store address of access
                                0073   389                 CONVERT_ACCVIO          ;  violation handler
     59    52    D0             0078   390         MOVL    R2,R9                   ; Store table address away
52   50    10    9C             007B   391         ROTL    #16,R0,R2               ; Store "dstlen" in R2
        0034    30             007F   392         BSBW    CVTPx_COMMON            ; Execute bulk as common code
                                0082   393
                                0082   394 ;+
                                0082   395 ; The common code routine returns here with the following relevant input.
                                0082   396 ;
                                0082   397 ;       R1      Address of least significant digit and sign of input string
                                0082   398 ;       R2      Number of digits in destination string (preserved across call)
                                0082   399 ;       R3      Address of last byte in destination to be loaded
                                0082   400 ;       R9      Address of 256-byte table (preserved across call)
                                0082   401 ;       R11     Saved PSW with condition codes to date (N=0,Z,V,C=0)
                                0082   402 ;
                                0082   403 ;       R4 is a scratch register
                                0082   404 ;
                                0082   405 ; The CVTPS instruction loads R8 in its initialization code. This instruction
                                0082   406 ; does not need R8 except at this time to determine the setting of the N-bit
                                0082   407 ; so R8 is loaded here. In addition, a check is required to insure that the
                                0082   408 ; Z-bit has its correct setting if the least significant digit is the first
                                0082   409 ; nonzero digit encountered in the input string.
                                0082   410 ;-
                                0082   411
          52    D5             0082   412         TSTL    R2                      ; Check for no remaining input
          11    13             0084   413         BEQL    10$                     ; Skip storing digit if nothing there
                                0086   414         MARK_POINT      CVTPT_ACCVIO
```

```
                     54    61    9A  0086    415          MOVZBL  (R1),R4                 ; Get last input digit
                                     0089    416          MARK_POINT      CVTPT_ACCVIO
                  63    6944    90  0089    417          MOVB    (R9)[R4],(R3)           ; Store associated destination byte
                                     008D    418          MARK_POINT      CVTPT_ACCVIO
               54    61    04    04  EF  008D    419          EXTZV   #4,#4,(R1),R4           ; Get least significant digit
                           03    13  0092    420          BEQL    10$                     ; Skip clearing Z-bit if zero
                     5B    04    8A  0094    421          BICB    #PSL$M_Z,R11            ; Clear saved Z-bit
                                     0097    422
                                     0097    423          MARK_POINT      CVTPT_ACCVIO
                58    61    F0 8F    8B  0097    424 10$:     BICB3   #^B11110000,(R1),R8     ; Sign "digit" to R8
                                     009C    425
                                     009C    426          CASE    R8,LIMIT=#10,TYPE=B,<-  ; Dispatch on sign
                                     009C    427                  30$,-                   ; 10 => +
                                     009C    428                  20$,-                   ; 11 => -
                                     009C    429                  30$,-                   ; 12 => +
                                     009C    430                  20$,-                   ; 13 => -
                                     009C    431                  30$,-                   ; 14 => +
                                     009C    432                  30$,-                   ; 15 => +
                                     009C    433                  >
                                     00AC    434
                  03 5B    02    E0  00AC    435 20$:     BBS     #PSL$V_Z,R11,30$        ; Skip if Z-bit set (negative zero)
                     5B    08    88  00B0    436          BISB    #PSL$M_N,R11            ; Set N-bit because sign is "-"
                        FF4A'    31  00B3    437 30$:     BRW     VAX$DECIMAL_EXIT        ; Exit through common code
```

```
                                00B6    439          .SUBTITLE       CVTPx_COMMON - Common Code / Packed to Numeric String
                                00B6    440 ;+
                                00B6    441 ; Functional Description:
                                00B6    442 ;
                                00B6    443 ;       This routine is used by both CVTPS and CVTPT to translate a packed
                                00B6    444 ;       decimal string of digits into its ASCII equivalent.
                                00B6    445 ;
                                00B6    446 ; Input Parameters:
                                00B6    447 ;
                                00B6    448 ;       R0 = srclen.rw          Length in digits of input decimal string
                                00B6    449 ;       R1 = srcaddr.ab         Address of input packed decimal string
                                00B6    450 ;       R2 = dstlen.rw          Number of digits in destination character string
                                00B6    451 ;       R3 = dstaddr.ab         Address of destination character string
                                00B6    452 ;
                                00B6    453 ;       (SP)    Address of instruction-specific completion code in CVTPS
                                00B6    454 ;               or CVTPT routine
                                00B6    455 ;
                                00B6    456 ; Implicit Input:
                                00B6    457 ;
                                00B6    458 ;       R10 must contain the address of an access violation handler in the
                                00B6    459 ;       event that any strings touched by this routine are not accessible.
                                00B6    460 ;
                                00B6    461 ; Output Parameters:
                                00B6    462 ;
                                00B6    463 ;       R0 = Size in digits of shorter of source and destination strings
                                00B6    464 ;       R1 = Address of least significant digit and sign of input string
                                00B6    465 ;       R2 = Number of digits in destination character string (unchanged)
                                00B6    466 ;       R3 = Address of last byte in destination to be loaded
                                00B6    467 ;
                                00B6    468 ;       R11 contains the partial condition codes accumulated by converting
                                00B6    469 ;               all but the least significant input digit
                                00B6    470 ;
                                00B6    471 ; Side Effects:
                                00B6    472 ;
                                00B6    473 ;       R4, R5, and R6 are used as scratch registers by this routine.
                                00B6    474 ;
                                00B6    475 ;       R7 through R10 are not used.
                                00B6    476 ;-
                                00B6    477
                                00B6    478          .ENABLE LOCAL_BLOCK
                                00B6    479
                                00B6    480 CVTPx_COMMON:
                                00B6    481          ROPRAND_CHECK   R0                 ; Insure that R0 LEQU 31
                                00C1    482          ROPRAND_CHECK   R2                 ; Insure that R2 LEQU 31
                          5B    00C9    483          MOVPSL  R11                        ; Get initial PSL
                    5B    04    00CB    484          INSV    #PSL$M_Z,#0,#4,R11         ; Set Z-bit, clear the rest
                    55    50    00D0    485          SUBL3   R2,R0,R5                   ; R5 is length difference
                          4F    00D4    486          BEQL    CVTPx_EQUAL                ; Life is easy if they're the same
                          44    00D6    487          BLSS    CVTPx_ZERO_FILL            ; Fill output with zeros if too large
                                00D8    488
                                00D8    489 ;+
                                00D8    490 ; **********                srclen GTRU dstlen                          **********
                                00D8    491 ;
                                00D8    492 ; The following code executes if the source string is larger than the
                                00D8    493 ; destination string. Excess high order input digits must be discarded. If
                                00D8    494 ; any of the input digits is not zero, then the V-bit is set in the saved
                                00D8    495 ; PSW (stored in R11). The low order digits will be moved as in the normal
```

```
                                00D8    496 ; case. A test for whether decimal overflow exceptions are to be generated
                                00D8    497 ; is made as part of final instruction processing.
                                00D8    498 ;
                                00D8    499 ;       R5 = R0 - R2  (R5 GTRU 0)
                                00D8    500 ;-
                                00D8    501
                                00D8    502 CVTPx_OVERFLOW_CHECK:
                          51    DD    00D8    503         PUSHL   R1                      ; Save initial input address
                       0B 50    E8    00DA    504         BLBS    R0,10$                  ; Skip single digit test for odd length
                                00DD    505         MARK_POINT      CVTPx_SAVED_R1
                54 61  04 00    EF    00DD    506         EXTZV   #0,#4,(R1),R4           ; First digit to R4 (Set R4<31:8> to 0)
                          18    12    00E2    507         BNEQ    40$                     ; Skip rest if nonzero. Nothing to be
                                00E4    508                                             ;  gained by hanging around.
                          55    D7    00E4    509         DECL    R5                      ; One less digit to check
                          51    D6    00E6    510         INCL    R1                      ; Point R1 to next byte in input string
             55 55 FF     8F    78    00E8    511 10$:    ASHL    #-1,R5,R5               ; Convert digit count to byte count
                          07    13    00ED    512         BEQL    30$                     ; Skip loop if no more double digits
                                00EF    513
                                00EF    514         MARK_POINT      CVTPx_SAVED_R1
                          81    95    00EF    515 20$:    TSTB    (R1)+                   ; Do rest two digits at a time
                          0C    12    00F1    516         BNEQ    40$                     ; Exit loop is nonzero digit
                       F9 55    F5    00F3    517         SOBGTR  R5,20$                  ; Check for end of loop
                                00F6    518
                       0C 52    E8    00F6    519 30$:    BLBS    R2,50$                  ; No lone digit if R2 odd
                                00F9    520         MARK_POINT      CVTPx_SAVED_R1
                61     F0 8F    93    00F9    521         BITB    #^B11110000,(R1)        ; Does upper nibble contain nonzero
                          06    13    00FD    522         BEQL    50$                     ; No, so we're all done
                                00FF    523
                                00FF    524 ; The following code executes if any of the discarded digits is nonzero. The
                                00FF    525 ; V-bit is set in the saved PSW, the saved Z-bit is cleared, and R1 is
                                00FF    526 ; updated to point to the remaining input string. The large comment at the
                                00FF    527 ; beginning of the module called VAX$DECIMAL explains why the INCL R5
                                00FF    528 ; instruction is necessary when R0, the length of the shorter string, is odd.
                                00FF    529
                   5B   04      8A    00FF    530 40$:    BICB    #PSL$M_Z,R11            ; Clear saved Z-bit
                   5B   02      88    0102    531         BISB    #PSL$M_V,R11            ; Set saved V-bit
              55   50   52      C3    0105    532 50$:    SUBL3   R2,R0,R5                ; Recompute difference (R5 GTRU 0)
                                0109    533
                                0109    534 ; The long comment at the beginning of the module explains the reason why
                                0109    535 ; we increment R5 when R2, the length of the shorter string, is odd.
                                0109    536
                     02 52      E9    0109    537         BLBC    R2,60$                  ; Need adjustment if odd output string
                          55    D6    010C    538         INCL    R5                      ; Adjust difference
             55 55    FF 8F      78    010E    539 60$:    ASHL    #-1,R5,R5               ; Convert digit count to byte count
          51    55      8E      C1    0113    540         ADDL3   (SP)+,R5,R1             ; "Restore" an updated input pointer
                50   52      D0    0117    541         MOVL    R2,R0                   ; Enter common code with updated
                                011A    542                                             ;  "srclen" equal to "dstlen"
                          09    11    011A    543         BRB     CVTPx_EQUAL             ; Join common code
                                011C    544
                                011C    545 ;+
                                011C    546 ; **********              srclen LSSU dstlen                          **********
                                011C    547 ;
                                011C    548 ; The following code executes if the destination string is longer than the
                                011C    549 ; source string. All excess digits in the destination string are filled
                                011C    550 ; with zero.
                                011C    551 ;-
                                011C    552
```

```
                     011C    553 CVTPx_ZERO_FILL:
                55   55  CE 011C    554         MNEGL   R5,R5                      ; Make digit count positive
                         011F    555
                         011F    556         MARK_POINT      CVTPx_BSBW
                83   30  90 011F    557 70$:    MOVB    #^A"0",(R3)+               ; Store a "0" in the output
                  FA 55  F5 0122    558         SOBGTR  R5,70$                     ; Check for end of loop
                         0125    559
                         0125    560 ;+
                         0125    561 ;**********      updated src:len EQL updated dstlen              **********
                         0125    562 ;
                         0125    563 ; The following code is a common meeting point for the three different input
                         0125    564 ; cases relating source length and destination length. Excess source or
                         0125    565 ; destination digits have already been dealt with. We are effectively
                         0125    566 ; dealing with input and output strings of equal length (as measured by
                         0125    567 ; number of digits).
                         0125    568 ;-
                         0125    569
                         0125    570 CVTPx_EQUAL:
                  18 50  E8 0125    571         BLBS    R0,90$                     ; No special first digit if R0 odd
                     50  D5 0128    572         TSTL    R0                         ; Also skip if no remaining digits
                     44  13 012A    573         BEQL    140$
                         012C    574         MARK_POINT      CVTPx_BSBW
            54   61  04  00  EF 012C    575         EXTZV   #0,#4,(R1),R4              ; First digit to R4 (Set R4<31:8> to 0)
                     03  13 0131    576         BEQL    80$                        ; Leave Z-bit alone if zero
                58   04  8A 0133    577         BICB    #PSL$M_Z,R11               ; Otherwise, clear Z-bit
                         0136    578
                         0136    579         MARK_POINT      CVTPx_BSBW
              83   FEC5 CF44  90 0136    580 80$:    MOVB    CVTPx_TABLE[R4],(R3)+      ; Move digit to output string
                     51  D6 013C    581         INCL    R1                         ; Advance input string pointer
                     50  D7 013E    582         DECL    R0                         ; One less digit to process
                         0140    583
            55   50   FF 8F  78 0140    584 90$:    ASHL    #-1,R0,R5                  ; Convert digit count to byte count
                     21  13 0145    585         BEQL    120$                       ; All done if zero
                         0147    586
                         0147    587         MARK_POINT      CVTPx_BSBW
                54   81  9A 0147    588 100$:   MOVZBL  (R1)+,R4                   ; Get next two input digits
                     1D  13 014A    589         BEQL    130$                       ; Step out of line if both are zero
                58   04  8A 014C    590         BICB    #PSL$M_Z,R11               ; Clear saved Z-bit
          56   54   04  04  EF 014F    591         EXTZV   #4,#4,R4,R6                ; Get high-order digit
                         0154    592         MARK_POINT      CVTPx_BSBW
              83   FEA7 CF46  90 0154    593         MOVB    CVTPx_TABLE[R6],(R3)+      ; Move associated character to output
            56   54   F0 8F  8B 015A    594         BICB3   #^B11110000,R4,R6          ; Get low-order digit
                         015F    595         MARK_POINT      CVTPx_BSBW
              83   FE9C CF46  90 015F    596         MOVB    CVTPx_TABLE[R6],(R3)+      ; Move associated character to output
                     DF 55  F5 0165    597 110$:   SOBGTR  R5,100$                    ; Test for end of loop
                         0168    598
                         0168    599 120$:   RSB                                ; Perform instruction-specific
                         0169    600                                            ;  end processing
                         0169    601
                         0169    602 ; This code is part of the main loop that moves input digits to the output
                         0169    603 ; string. This code only executes when a digit pair consisting of two zeros
                         0169    604 ; is detected. Note that this is an optimization that recognizes that the
                         0169    605 ; individual digits do not have to be translated in order to load the
                         0169    606 ; destination string.
                         0169    607
                         0169    608         MARK_POINT      CVTPx_BSBW
                83   3030 8F  B0 0169    609 130$:   MOVW    #^A"00",(R3)+              ; Move the pair to the output
```

```
                                                K 16
                     F5   11  016E   610          BRB     110$                    ; Rejoin at the end of the loop
                              0170   611
                              0170   612 ; We have advanced too far in the destination string. Back up by one byte
                              0170   613 ; and let the caller correctly load the final output byte.
                              0170   614
                     53   D7  0170   615 140$:    DECL    R3
                          05  0172   616          RSB
                              0173   617
                              0173   618          .DISABLE        LOCAL_BLOCK
```

```
                        .SUBTITLE       VAX$CVTxP - Convert Numeric String to Packed
;+
; Functional Description:
;
;       The conversion from a numeric string to a packed decimal string
;       (CVTSP and CVTTP instructions) consists of much common code and
;       two small pieces of code that are instruction specific, the
;       beginning and a portion of the end processing. The actual routine
;       exit path is the common exit path from this module, VAX$DECIMAL_EXIT.
;
;       The two routines perform instruction-specific operations on the
;       first byte in the stream. The bulk of the work is done by a common
;       subroutine. Some instruction-specific end processing is done before
;       final control is passed to VAX$DECIMAL_EXIT.
;
;       The structure is something like the following.
;
;               CVTSP                                   CVTTP
;               -----                                   -----
;
;       Skip over sign character                Store table address
;                                               Unpack registers
;                                                       |
;                                                      /
;                                                     /
;                                                    |
;                                                    v
;                               Handle unequal srclen and dstlen
;                               Move all digits except last digit
;                                                    |
;                                                   /
;                                                  /
;                                                 |
;                                                 v
;       Move last digit to output               Use table to move last digit
;       Move sign to output                     and sign to output string
;                                                       |
;                                                      /
;                                                     /
;                                                    |
;                                                    v
;                                       VAX$DECIMAL_EXIT
;                               Set condition codes and registers
;                               to their final values
;
; Input Parameters:
```

```
                     0173   677 ;
                     0173   678 ;       See instruction-specific entry points
                     0173   679 ;
                     0173   680 ; Output Parameters:
                     0173   681 ;
                     0173   682 ;       R0 = 0
                     0173   683 ;       R1 = Address of lowest addressed byte of destination string
                     0173   684 ;            (See instruction-specific header for details)
                     0173   685 ;       R2 = 0
                     0173   686 ;       R3 = Address of byte containing most significant digit of
                     0173   687 ;            the source string
                     0173   688 ;
                     0173   689 ; Condition Codes:
                     0173   690 ;
                     0173   691 ;       N <- destination string LSS 0
                     0173   692 ;       Z <- destination string EQL 0
                     0173   693 ;       V <- decimal overflow
                     0173   694 ;       C <- 0
                     0173   695 ;
                     0173   696 ; Notes:
                     0173   697 ;
                     0173   698 ;       Both of these instructions check the input strings for legal decimal
                     0173   699 ;       digits. If a character other than the ASCII representation of a
                     0173   700 ;       decimal digit is detected in the input string, a reserved operand
                     0173   701 ;       abort is generated. This exception is not restartable.
                     0173   702 ;
                     0173   703 ;       In addition, the CVTSP instruction insures that the sign character is
                     0173   704 ;       one of "+", " ", or "-".
                     0173   705 ;
                     0173   706 ;       The CVTTP instruction uses the highest addressed byte as an offset
                     0173   707 ;       into a 256-byte table. The byte that is retrieved from this table is
                     0173   708 ;       checked to determine that its high nibble contains a legal decimal
                     0173   709 ;       digit and its low nibble contains a legal sign.
                     0173   710 ;-
                     0173   711
                     0173   712         .SUBTITLE       Data Declarations / Numeric String to Packed
                     0173   713
                     0173   714 ;+
                     0173   715 ; The following tables contains the decimal equivalents of the ten decimal
                     0173   716 ; digits. One table is used if the low nibble of a byte is being loaded
                     0173   717 ; (an even numbered digit). The other table is used when the high nibble
                     0173   718 ; of a byte is being loaded (odd numbered digit).
                     0173   719 ;-
                     0173   720
                     0173   721 ; Table for entry into low order nibble
                     0173   722
                     0173   723 CVTxP_TABLE_LOW:
04 03 02 01 00       0173   724         .BYTE   ^X00 , ^X01 , ^X02 , ^X03 , ^X04
09 08 07 06 05       0178   725         .BYTE   ^X05 , ^X06 , ^X07 , ^X08 , ^X09
                     017D   726
                     017D   727 ; Table for entry into high order nibble
                     017D   728
                     017D   729 CVTxP_TABLE_HIGH:
40 30 20 10 00       017D   730         .BYTE   ^X00 , ^X10 , ^X20 , ^X30 , ^X40
90 80 70 60 50       0182   731         .BYTE   ^X50 , ^X60 , ^X70 , ^X80 , ^X90
```

```
                     0187   733          .SUBTITLE       VAX$CVTSP - Convert Leading Separate Numeric to Packed
                     0187   734 ;+
                     0187   735 ; Functional Description:
                     0187   736 ;
                     0187   737 ;       The source numeric string specified by  the  source  length  and  source
                     0187   738 ;       address  operands  is  converted  to  a  packed  decimal  string and the
                     0187   739 ;       destination string specified by the destination address and  destination
                     0187   740 ;       length operands is replaced by the result.
                     0187   741 ;
                     0187   742 ; Input Parameters:
                     0187   743 ;
                     0187   744 ;       R0 = srclen.rw          Number of digits in source character string
                     0187   745 ;       R1 = srcaddr.ab         Address of input character string
                     0187   746 ;       R2 = dstlen.rw          Length in digits of output decimal string
                     0187   747 ;       R3 = dstaddr.ab         Address of destination packed decimal string
                     0187   748 ;
                     0187   749 ; Output Parameters:
                     0187   750 ;
                     0187   751 ;       R0 = 0
                     0187   752 ;       R1 = Address of the sign byte of the source string
                     0187   753 ;       R2 = 0
                     0187   754 ;       R3 = Address of byte containing most significant digit of
                     0187   755 ;            the destination string
                     0187   756 ;
                     0187   757 ; Condition Codes:
                     0187   758 ;
                     0187   759 ;       N <- destination string LSS 0
                     0187   760 ;       Z <- destination string EQL 0
                     0187   761 ;       V <- decimal overflow
                     0187   762 ;       C <- 0
                     0187   763 ;
                     0187   764 ; Notes:
                     0187   765 ;
                     0187   766 ;       Note that the two entry points VAX$CVTSP and VAX$CVTTP must save the
                     0187   767 ;       exact same set of registers because the two routines use a common exit
                     0187   768 ;       path that includes a POPR instruction that restores registers. In
                     0187   769 ;       fact, by saving all registers, even if one or two of them are not
                     0187   770 ;       needed, we can use the common exit path from this module.
                     0187   771 ;-
                     0187   772
                     0187   773 VAX$CVTSP::
 0FFF 8F     BB      0187   774          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>      ; Save the lot
       51    D6      0188   775          INCL    R1                              ; Skip byte containing sign for now
     00E7    30      018D   776          BSBW    CVTxP_COMMON                    ; Execute bulk as common code
                     0190   777
                     0190   778 ;+
                     0190   779 ; The common code routine returns here with the following relevant input.
                     0190   780 ;
                     0190   781 ;       R0      Number of digits remaining in source and destination strings
                     0190   782 ;       R1      Address of last (highest addressed) byte in source string
                     0190   783 ;       R3      Address of least significant digit and sign of output string
                     0190   784 ;       R4      R4<31:8> must be zero on input to this routine
                     0190   785 ;       R11     Saved PSW with condition codes to date (N=0,Z,V,C=0)
                     0190   786 ;
                     0190   787 ;       CVTSP_A_SRCADDR(SP)     Saved R1 at input, address of sign character
                     0190   788 ;
                     0190   789 ;       R4 is a scratch register
```

```
                     0190   790 ;
                     0190   791 ; The last input digit is moved to the output stream, after a check that it
                     0190   792 ; represents a legal decimal digit. A check is also required to insure that
                     0190   793 ; the Z-bit has its correct setting if this digit is the first nonzero digit
                     0190   794 ; encountered in the input string. The sign of the input string is checked
                     0190   795 ; for a legal value and transformed into one of two legal output signs, 12
                     0190   796 ; for "+" and 13 for "-".
                     0190   797 ;-
                     0190   798
                     0190   799          MARK_POINT      CVTSP_ACCVIO
          63   0C    90  0190   800          MOVB    #12,(R3)                ; Assume that sign is plus
               50    D5  0193   801          TSTL    R0                      ; Check for zero length input string
               15    13  0195   802          BEQL    20$                     ; Skip storing digit if nothing left
                     0197   803          MARK_POINT      CVTSP_ACCVIO
     54   61   30    83  0197   804          SUBB3   #^A"0",(R1),R4          ; Get least significant digit
               22    1F  019B   805          BLSSU   30$                     ; Reserved operand if not a digit
               03    13  019D   806          BEQL    10$                     ; Skip clearing Z-bit if zero
          5B   04    8A  019F   807          BICB    #PSL$M_Z,R11            ; Clear saved Z-bit
          09   54    91  01A2   808 10$:     CMPB    R4,#9                   ; Check digit against top of range
               18    1A  01A5   809          BGTRU   30$                     ; Reserved operand if over the top
                     01A7   810          MARK_POINT      CVTSP_ACCVIO
     63   D2 AF44    80  01A7   811          ADDB    CVTxP_TABLE_HIGH[R4],(R3)       ; Store final output digit
                     01AC   812
                     01AC   813          MARK_POINT      CVTSP_ACCVIO
       54   04 BE    9A  01AC   814 20$:     MOVZBL  @CVTSP_A_SRCADDR(SP),R4 ; Get sign character from input string
                     01B0   815
                     01B0   816          CASE    R4,LIMIT=#^A"+",TYPE=B,<-       ; Dispatch on sign character
                     01B0   817                  50$,-                   ; Character is "+"
                     01B0   818                  30$,-                   ; Sign character is ";" (illegal input)
                     01B0   819                  40$,-                   ; Character is "-"
                     01B0   820                  >
                     01BA   821
     20   54    91  01BA   822          CMPB    R4,#^A" "               ; Blank is also legal "plus sign"
               15    13  01BD   823          BEQL    50$
                     01BF   824
                     01BF   825 ; Error path for all code paths that detect an illegal character in
                     01BF   826 ; the input stream
                     01BF   827
          0151     31  01BF   828 30$:     BRW     DECIMAL_ROPRAND_NO_PC   ; Reserved operand abort on illegal input
                     01C2   829
                     01C2   830 ; The sign of the input stream was "-". If something other than negative
                     01C2   831 ; zero, set the N-bit and adjust the sign.
                     01C2   832
          5B   08    88  01C2   833 40$:     BISB    #PSL$M_N,R11            ; Set N-bit because sign is "-"
                     01C5   834          MARK_POINT      CVTSP_ACCVIO
               63    96  01C5   835          INCB    (R3)                    ; Change sign from "+" (12) to "-" (13)
     09   5B   02    E1  01C7   836          BBC     #PSL$V_Z,R11,50$        ; All done unless negative zero
          5B   08    8A  01CB   837          BICB    #PSL$M_N,R11            ; Clear the saved N-bit
     02   5B   01    E0  01CE   838          BBS     #PSL$V_V,R11,50$        ; The output sign is ignored if overflow
                     01D2   839          MARK_POINT      CVTSP_ACCVIO
               63    97  01D2   840          DECB    (R3)                    ; Change sign back so -0 becomes +0
            FE29'    31  01D4   841 50$:     BRW     VAX$DECIMAL_EXIT        ; Exit through common code
```

```
01D7   843          .SUBTITLE       VAX$CVTTP - Convert Trailing Numeric to Packed
01D7   844 ;+
01D7   845 ; Functional Description:
01D7   846 ;
01D7   847 ;       The source trailing numeric string specified by the  source  length  and
01D7   848 ;       source  address operands is converted to a packed decimal string and the
01D7   849 ;       destination packed decimal string specified by the  destination  address
01D7   850 ;       and destination length operands is replaced by the result.
01D7   851 ;
01D7   852 ;       Conversion is effected by using the highest addressed (trailing) byte of
01D7   853 ;       the  source  string  as  an  unsigned  index into a 256 byte table whose
01D7   854 ;       zeroth entry is specified by the table address operand.  The  byte  read
01D7   855 ;       out  of the table replaces the highest addressed byte of the destination
01D7   856 ;       string (i.e.  the byte containing the sign  and  the  least  significant
01D7   857 ;       digit).  The  remaining  packed  digits  of  the destination string are
01D7   858 ;       replaced by the low order 4 bits  of  the  corresponding  bytes  in  the
01D7   859 ;       source string.
01D7   860 ;
01D7   861 ; Input Parameters:
01D7   862 ;
01D7   863 ;       R0 <15:0>  = srclen.rw  Number of digits in source character string
01D7   864 ;       R0 <31:16> = dstlen.rw  Length in digits of output decimal string
01D7   865 ;       R1         = srcaddr.ab Address of input character string
01D7   866 ;       R2         = tbladdr.ab Address of 256-byte table used for sign conversion
01D7   867 ;       R3         = dstaddr.ab Address of destination packed decimal string
01D7   868 ;
01D7   869 ; Output Parameters:
01D7   870 ;
01D7   871 ;       R0 = 0
01D7   872 ;       R1 = Address of most significant digit of the source string
01D7   873 ;       R2 = 0
01D7   874 ;       R3 = Address of byte containing most significant digit of
01D7   875 ;            the destination string
01D7   876 ;
01D7   877 ; Condition Codes:
01D7   878 ;
01D7   879 ;       N <- destination string LSS 0
01D7   880 ;       Z <- destination string EQL 0
01D7   881 ;       V <- decimal overflow
01D7   882 ;       C <- 0
01D7   883 ;
01D7   884 ; Notes:
01D7   885 ;
01D7   886 ;       1.  Note that the two entry points VAX$CVTSP and VAX$CVTTP must save the
01D7   887 ;           exact same set of registers because the two routines use a common exit
01D7   888 ;           path that includes a POPR instruction that restores registers. In
01D7   889 ;           fact, by saving all registers, even if one or two of them are not
01D7   890 ;           needed, we can use the common exit path from this module.
01D7   891 ;
01D7   892 ;       2.  See the routine header for CVTPT for an explanation of the _JSB and
01D7   893 ;           _RESTART entry points.
01D7   894 ;-
01D7   895
01D7   896 ; There is a single case where the common subroutine cannot be used. If the
01D7   897 ; output length is zero, then the final character in the input string would
01D7   898 ; be subjected to the rather stringent legality test that it lie between
01D7   899 ; ASCII 0 and ASCII 9. In fact, it is the translated character that must be
```

```
                                01D7   900 ; tested. There are three cases.
                                01D7   901 ;
                                01D7   902 ;       The input length is also zero. In this case, the common code path can
                                01D7   903 ;       be used because the input and output length are equal. (In fact, the
                                01D7   904 ;       subroutine does little more than set the condition codes and load
                                01D7   905 ;       registers.
                                01D7   906 ;
                                01D7   907 ;       The input consists of a single character. In this case, this single
                                01D7   908 ;       character is translated and tested for legality. Note that the
                                01D7   909 ;       subroutine is also called here to set condition codes and the like.
                                01D7   910 ;
                                01D7   911 ;       The input size is larger than one. In this case, the common subroutine
                                01D7   912 ;       is called with the input size reduced by one. The leading characters
                                01D7   913 ;       are tested by the subroutine which returns here to allow the final
                                01D7   914 ;       character to be tested.
                                01D7   915 ;
                                01D7   916 ; Note that this is not a commonly travelled code path so that the seemingly
                                01D7   917 ; excessive amount of code necessary to achieve accuracy is not a performance
                                01D7   918 ; problem.
                                01D7   919
                                01D7   920         .ENABLE LOCAL_BLOCK
                                01D7   921
                                01D7   922 1$:     ROPRAND_CHECK   R0                      ; Insure that R0 LEQU 31
                          3C 13 01E2   923         BEQL    5$                              ; Back in line if source length zero
                          50 D7 01E4   924         DECL    R0                              ; Reduce input length by one
                      008E   30 01E6   925         BSBW    CVTxP_COMMON                    ; Check leading digits for legality
                                01E9   926         MARK_POINT      CVTTP_ACCVIO
                       54 61   9A 01E9   927         MOVZBL  (R1),R4                         ; Get last input byte
                                01EC   928         MARK_POINT      CVTTP_ACCVIO
                       54 6944 9A 01EC   929         MOVZBL  (R9)[R4],R4                     ; Get associated output byte from table
                                01F0   930         MARK_POINT      CVTTP_ACCVIO
                  63 54 F0 8F   8B 01F0   931         BICB3   #^B11110000,R4,(R3)             ; Only store sign in output string
               50 54 04 04      EF 01F5   932         EXTZV   #4,#4,R4,R0                     ; Get low-order digit
                        44      13 01FA   933         BEQL    10$                             ; Join exit code if zero
                     5B 02      B8 01FC   934         BISB2   #PSL$M_V,R11                    ; Set V-bit in saved PSW
                     09 50      D1 01FF   935         CMPL    R0,#9                           ; Is the digit within range?
                        3C      1B 0202   936         BLEQU   10$                             ; Yes, join the exit code
                      010C      31 0204   937         BRW     DECIMAL_ROPRAND_NO_PC           ; Otherwise, report exception
                                0207   938
                                0207   939 VAX$CVTTP_JSB::
                  07 50   0F    E5 0207   940         BBCC    #CVTTP_V_FPD,R0,VAX$CVTTP       ; Have we been here before?
                                020B   941
                                020B   942         ASSUME  CVTTP_B_DELTA_PC EQ 3           ; Make sure that we clear the right byte
                                020B   943
                                020B   944 VAX$CVTTP_RESTART::
               50 FF000000 8F   CA 020B   945         BICL2   #^XFF000000,R0                  ; Eliminate delta-PC from "dstlen"
                                0212   946
                                0212   947 VAX$CVTTP::
                     0FFF 8F    BB 0212   948         PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>      ; Save the lot
                        59 52   D0 0216   949         MOVL    R2,R9                           ; Store table address away
            52 50 10    10      EF 0219   950         EXTZV   #16,#16,R0,R2                   ; Store "dstlen" in R2
                        B7      13 021E   951         BEQL    1$                              ; Perform extraordinary check if zero
                      0054      30 0220   952 5$:     BSBW    CVTxP_COMMON                    ; Execute bulk as common code
                                0223   953
                                0223   954 ;+
                                0223   955 ; The common code routine returns here with the following relevant input.
                                0223   956 ;
```

```
                                0223   957 ;       R0      Number of digits remaining in source and destination strings
                                0223   958 ;       R1      Address of last (highest addressed) byte in source string
                                0223   959 ;       R3      Address of least significant digit and sign of output string
                                0223   960 ;       R9      Address of 256-byte table (preserved across call)
                                0223   961 ;       R11     Saved PSW with condition codes to date (N=0,Z,V,C=0)
                                0223   962 ;
                                0223   963 ;       R4 is a scratch register
                                0223   964 ;
                                0223   965 ; The last byte of the input string is used as an index into the 256-byte
                                0223   966 ; table that contains the last output byte. The contents of this byte are
                                0223   967 ; tested for a legal decimal digit in its upper nibble and a legal sign
                                0223   968 ; representation (10 through 15) in its low nibble. The Z-bit is cleared
                                0223   969 ; if the digit is 1 through 9 to cover the case that this is the first
                                0223   970 ; nonzero digit in the input string.
                                0223   971 ;-
                                0223   972
                          50    D5  0223   973         TSTL    R0                      ; Check for no remaining input
                          4A    13  0225   974         BEQL    70$                     ; Special case if input length now zero
                                0227   975         MARK_POINT      CVTTP_ACCVIO
                    54    61    9A  0227   976         MOVZBL  (R1),R4                 ; Get last input byte
                                022A   977         MARK_POINT      CVTTP_ACCVIO
                54    6944    9A  022A   978         MOVZBL  (R9)[R4],R4             ; Get associated output byte from table
                                022E   979         MARK_POINT      CVTTP_ACCVIO
                63    54      90  022E   980         MOVB    R4,(R3)                 ; Store in destination string
          50    54    04      04  EF  0231   981         EXTZV   #4,#4,R4,R0             ; Get least significant digit
                      08      13  0236   982         BEQL    10$                     ; Skip clearing Z-bit if zero
                09    50      91  0238   983         CMPB    R0,#9                   ; Check for legal range
                      18      14  023B   984         BGTR    20$                     ; Reserved operand if 10 through 15
                5B    04      8A  023D   985         BICB    #PSL$M_Z,R11            ; Clear saved Z-bit
       50   54    F0 8F       88  0240   986 10$:     BICB3   #^B11110000,R4,R0       ; Sign "digit" to R0
                                0245   987
                                0245   988         CASE    R0,LIMIT=#10,TYPE=B,<-  ; Dispatch on sign
                                0245   989                 50$,-                   ; 10 => +
                                0245   990                 30$,-                   ; 11 => -
                                0245   991                 60$,-                   ; 12 => +
                                0245   992                 40$,-                   ; 13 => -
                                0245   993                 50$,-                   ; 14 => +
                                0245   994                 50$,-                   ; 15 => +
                                0245   995                 >
                                0255   996
                      00BB      31  0255   997 20$:     BRW     DECIMAL_ROPRAND_NO_PC   ; Reserved operand if sign is 0 to 9
                                0258   998
                                0258   999 ; A minus sign of 11 must be changed to 13, the preferred minus representation
                                0258  1000
                                0258  1001         MARK_POINT      CVTTP_ACCVIO
                63    02      80  0258  1002 30$:     ADDB2   #2,(R3)                 ; Change 11 to 13, preferred minus sign
                5B    08      88  025B  1003 40$:     BISB    #PSL$M_N,R11            ; Set N-bit because sign is "-"
          0C    5B    02      E1  025E  1004         BBC     #PSL$V_Z,R11,60$        ; All done unless negative zero
                5B    08      8A  0262  1005         BICB    #PSL$M_N,R11            ; Clear the saved N-bit
          05    5B    01      E0  0265  1006         BBS     #PSL$V_V,R11,60$        ; The output sign is ignored if overflow
                                0269  1007
                                0269  1008 ; If the sign character is a 10, 14, or 15, it must be changed to a 12, the
                                0269  1009 ; preferred plus sign before joining the exit code.
                                0269  1010
                                0269  1011         MARK_POINT      CVTTP_ACCVIO
    63    04    00    0C      F0  0269  1012 50$:     INSV    #12,#0,#4,(R3)          ; Store a 12 as the output sign
                      FD8F'     31  026E  1013 60$:     BRW     VAX$DECIMAL_EXIT        ; Exit through common code
```

```
                     0271  1014
                     0271  1015 ; If the source string has zero length, the destination is set identically
                     0271  1016 ; to zero. The following instruction sequence assumes that the Z-bit was
                     0271  1017 ; set in the initialization code for this routine.
                     0271  1018
                     0271  1019          MARK_POINT      CVTTP_ACCVIO
63   0C   90        0271  1020 70$:      MOVB    #12,(R3)                ; Store "+" in output string
   FD89'  31         0274  1021          BRW     VAX$DECIMAL_EXIT        ; Exit through common code
                     0277  1022
                     0277  1023          .DISABLE        LOCAL_BLOCK
```

```
                                0277  1025          .SUBTITLE       CVTxP_COMMON - Common Code / Numeric String to Packed
                                0277  1026 ;+
                                0277  1027 ; Functional Description:
                                0277  1028 ;
                                0277  1029 ;       This routine is shared by both CVTSP and CVTPT to translate an ASCII
                                0277  1030 ;       string that contains only the characters "0" to "9" into an equivalent
                                0277  1031 ;       packed decimal string. A check is made for legal input digits and a
                                0277  1032 ;       reserved operand exception generated if an illegal digit is
                                0277  1033 ;       encountered.
                                0277  1034 ;
                                0277  1035 ; Input Parameters:
                                0277  1036 ;
                                0277  1037 ;       R0 = srclen.rw          Number of digits in source character string
                                0277  1038 ;       R1 = srcaddr.ab         Address of first digit in input character string
                                0277  1039 ;       R2 = dstlen.rw          Length in digits of output decimal string
                                0277  1040 ;       R3 = dstaddr.ab         Address of destination packed decimal string
                                0277  1041 ;
                                0277  1042 ;       (SP)    Address of instruction-specific completion code in CVTSP
                                0277  1043 ;               or CVTTP routine
                                0277  1044 ;
                                0277  1045 ; Output Parameters:
                                0277  1046 ;
                                0277  1047 ;       R0 = Size in digits of shorter of source and destination strings
                                0277  1048 ;       R1 = Address of lowest addressed byte of source string
                                0277  1049 ;            (See instruction-specific header for details)
                                0277  1050 ;       R2 = Number of digits in destination packed decimal string
                                0277  1051 ;       R3 = Address of byte containing most significant digit of
                                0277  1052 ;            the destination string
                                0277  1053 ;
                                0277  1054 ;       R11 contains the partial condition codes accumulated by converting
                                0277  1055 ;               all but the least significant input digit
                                0277  1056 ;
                                0277  1057 ; Implicit Output:
                                0277  1058 ;
                                0277  1059 ;       R4<31:8> is zero to insure that CVTSP works correctly
                                0277  1060 ;
                                0277  1061 ;       R10 is loaded with the address of an access violation handler in the
                                0277  1062 ;       event that any strings touched by this routine are not accessible.
                                0277  1063 ;
                                0277  1064 ; Side Effects:
                                0277  1065 ;
                                0277  1066 ;       R4 and R5 are used as scratch registers by this routine.
                                0277  1067 ;
                                0277  1068 ;       R6 through R9 are not used.
                                0277  1069 ;-
                                0277  1070
                                0277  1071          .ENABLE LOCAL_BLOCK
                                0277  1072
                                0277  1073 CVTxP_COMMON:
                                0277  1074          ROPRAND_CHECK   R0              ; Insure that R0 LEQU 31
                                0282  1075          ROPRAND_CHECK   R2              ; Insure that R2 LEQU 31
                          5B    028A  1076          MOVPSL  R11                     ; Get initial PSL
                    5B    04    028C  1077          INSV    #PSL$M_Z,#0,#4,R11      ; Set Z-bit, clear the rest
                                0291  1078          ESTABLISH_HANDLER       -       ; Store address of access
                                0291  1079                  CONVERT_ACCVIO          ;  violation handler
                    55    50    0296  1080          SUBL3   R2,R0,R5                ; R5 is length difference
                          38    029A  1081          BEQL    CVTxP_EQUAL             ; Life is easy if they're the same
```

```
                 1B   19  029C  1082          BLSS    CVTxP_ZERO_FILL         ; Fill output with zeros if its too large
                         029E  1083
                         029E  1084 ;+
                         029E  1085 ; **********              srclen GTRU dstlen                          **********
                         029E  1086 ;
                         029E  1087 ; The following code executes if the source string is larger than the
                         029E  1088 ; destination string. Excess high order input digits must be discarded. If
                         029E  1089 ; any of the input digits is not zero, then the V-bit is set in the saved PSW
                         029E  1090 ; (stored in R11). In addition, digits must be checked for legal values
                         029E  1091 ; (ASCII 0 through ASCII 9) before they are discarded in order to determine
                         029E  1092 ; whether to generate a reserved operand abort. The low order digits will be
                         029E  1093 ; moved as in the normal case. A test for whether decimal overflow exceptions
                         029E  1094 ; are to be generated is made as part of final instruction processing.
                         029E  1095 ;
                         029E  1096 ;       R5 = R0 - R2  (R5 GTRU 0)
                         029E  1097 ;-
                         029E  1098
                         029E  1099 CVTxP_OVERFLOW_CHECK:
                         029E  1100          MARK_POINT      CVTxP_BSBW
                 30   81  91  029E  1101 10$:     CMPB    (R1)+,#^A"0"            ; Is digit ASCII zero?
                      08  12  02A1  1102          BNEQ    30$                     ; Exit loop if other than zero
                   F8 55  F5  02A3  1103 20$:     SOBGTR  R5,10$                  ; Test for more excess digits
                         02A6  1104
                 50   52  D0  02A6  1105          MOVL    R2,R0                   ; Update input length for skipped digits
                      29  11  02A9  1106          BRB     CVTxP_EQUAL             ; Join common code
                         02AB  1107
                         02AB  1108 ; The following code executes if any of the discarded digits is nonzero.
                         02AB  1109 ; If the digit is the ASCII representation of a decimal digit, then the
                         02AB  1110 ; V-bit is set in the saved PSW and the saved Z-bit is cleared. The loop
                         02AB  1111 ; is reentered where we left it to continue the search for legal input
                         02AB  1112 ; digits. (Note that this is different from the CVTPx case where, once an
                         02AB  1113 ; overflow was detected, the remaining excess input digits could be skipped.)
                         02AB  1114
                      09  1F  02AB  1115 30$:     BLSSU   40$                     ; Reserved operand if outside range
                   5B 02  88  02AD  1116          BISB    #PSL$M_V,R11            ; Set saved V-bit
                         02B0  1117          MARK_POINT      CVTxP_BSBW
                 39   FF A1  91  02B0  1118          CMPB    -1(R1),#^A"9"           ; Compare digit to ASCII 9
                      ED  1B  02B4  1119          BLEQU   20$                     ; Back in loop if inside range
                    0057  31  02B6  1120 40$:     BRW     DECIMAL_ROPRAND         ; Signal illegal digit abort
                         02B9  1121
                         02B9  1122 ;+
                         02B9  1123 ; **********              srclen LSSU dstlen                          **********
                         02B9  1124 ;
                         02B9  1125 ; The following code executes if the destination string is longer than the
                         02B9  1126 ; source string. All excess digits in the destination string are filled
                         02B9  1127 ; with zero.
                         02B9  1128 ;-
                         02B9  1129
                         02B9  1130 CVTxP_ZERO_FILL:
                   55 55  CE  02B9  1131          MNEGL   R5,R5                   ; Make digit count positive
                         02BC  1132
                   09 50  E9  02BC  1133          BLBC    R0,50$                  ; Different code paths for even and odd
                         02BF  1134                                                 ;  input string sizes (the shorter one)
                         02BF  1135
                         02BF  1136 ; Shorter string has odd number of digits. Note that the divide by two can
                         02BF  1137 ; never produce zero because R5 is always nonzero before the INCL so that R5
                         02BF  1138 ; is always at least two before the divide takes place. The comment at the
```

```
                                02BF  1139 ; beginning of the module explains the two different code paths based on the
                                02BF  1140 ; parity of the input (shorter) string.
                                02BF  1141
                          55    D6  02BF  1142          INCL    R5                      ; Adjust before divide by two
                    55    55  04  01  EF  02C1  1143          EXTZV   #1,#4,R5,R5             ; Convert digit count to byte count
                                07  11  02C6  1144          BRB     60$                     ; Join common loop
                                02C8  1145
                                02C8  1146 ; Shorter string has an even number of digits.
                                02C8  1147
                    55    55  04  01  EF  02C8  1148 50$:    EXTZV   #1,#4,R5,R5             ; Convert digit count to byte count
                                05  13  02CD  1149          BEQL    CVTxP_EQUAL             ; No loop if byte count is zero
                                02CF  1150
                                02CF  1151          MARK_POINT      CVTxP_BSBW
                          83    94  02CF  1152 60$:    CLRB    (R3)+                   ; Store a pair of zeros in output string
                       FB 55    F5  02D1  1153          SOBGTR  R5,60$                  ; Test for more bytes to clear
                                02D4  1154
                                02D4  1155 ;+
                                02D4  1156 ;**********      updated srclen EQL updated dstlen               **********
                                02D4  1157 ;
                                02D4  1158 ; The following code is a common meeting point for the three different input
                                02D4  1159 ; cases relating source length and destination length. Excess source or
                                02D4  1160 ; destination digits have already been dealt with. We are effectively
                                02D4  1161 ; dealing with input and output strings of equal length (as measured by
                                02D4  1162 ; number of digits).
                                02D4  1163 ;-
                                02D4  1164
                                02D4  1165 CVTxP_EQUAL:
                          54    D4  02D4  1166          CLRL    R4                      ; Insure that R4<31:8> is zero
                    55    50  04  01  EF  02D6  1167          EXTZV   #1,#4,R0,R5             ; Convert digit count to byte count
                                32  13  02DB  1168          BEQL    110$                    ; Down to last digit if zero
                                02DD  1169
                                02DD  1170 ; If the count of remaining digits is even, we need to jump into the middle
                                02DD  1171 ; of the loop. But the store operation in the second half of the loop uses a
                                02DD  1172 ; BISB2, assuming that the high order nibble is already cleared (which it is if
                                02DD  1173 ; we also execute the first half of the loop). In order to insure that the high
                                02DD  1174 ; order nibble has a zero stored in it, we jump to the last instruction of the
                                02DD  1175 ; first half of the loop. Because we just cleared R4, the MOVB instruction at
                                02DD  1176 ; 90$ stores a zero in the appropriate byte of the output string.
                                02DD  1177
                       10 50    E9  02DD  1178          BLBC    R0,90$                  ; To middle of loop if digit count even
                                02E0  1179
                                02E0  1180          MARK_POINT      CVTxP_BSBW
                    54  81  30  83  02E0  1181 70$:    SUBB3   #^A"0",(R1)+,R4         ; Convert ASCII to digit
                                D0  1F  02E4  1182          BLSSU   40$                     ; Abort instruction if out of range
                                03  13  02E6  1183          BEQL    80$                     ; Do not clear Z-bit if digit is zero
                          5B    04  8A  02E8  1184          BICB    #PSL$M_Z,R11            ; Clear Z-bit when digit is 1 to 9
                          09    54  91  02EB  1185 80$:    CMPB    R4,#9                   ; Check for other end of range
                                C6  1A  02EE  1186          BGTRU   40$                     ; Abort if outside the other end, too
                                02F0  1187
                                02F0  1188          MARK_POINT      CVTxP_BSBW
                63  FE88 CF44   90  02F0  1189 90$:    MOVB    CVTxP_TABLE_HIGH[R4],(R3)       ; Store digit in high nibble
                                02F6  1190
                                02F6  1191 ; Note that the above instruction also clears out the low order four bits in
                                02F6  1192 ; the currently addressed byte in the output packed decimal string.
                                02F6  1193
                                02F6  1194          MARK_POINT      CVTxP_BSBW
                    54  81  30  83  02F6  1195          SUBB3   #^A"0",(R1)+,R4         ; Convert ASCII to digit
```

```
                      BA   1F  02FA  1196              BLSSU   40$                         ; Abort instruction if out of range
                      03   13  02FC  1197              BEQL    100$                        ; Do not clear Z-bit if digit is zero
                   5B 04   8A  02FE  1198              BICB    #PSL$M_Z,R11                ; Clear Z-bit when digit is 1 to 9
                   09 54   91  0301  1199  100$:      CMPB    R4,#9                       ; Check for other end of range
                      B0   1A  0304  1200              BGTRU   40$                         ; Abort if outside the other end, too
                           0306  1201              MARK_POINT      CVTxP_BSBW
                83 FE68 CF44   88  0306  1202              BISB2   CVTxP_TABLE_LOW[R4],(R3)+       ; Store digit in low nibble
                           030C  1203
                      D1 55    F5  030C  1204              SOBGTR  R5,70$                  ; Test for end of loop
                           030F  1205
                      05       05  030F  1206  110$:      RSB                             ; Perform instruction-specific
                           0310  1207                                                      ;  end processing
                           0310  1208
                           0310  1209              .DISABLE        LOCAL_BLOCK
```

```
                                L 1
0310  1211          .SUBTITLE       DECIMAL_ROPRAND
0310  1212 ;-
0310  1213 ; Functional Description:
0310  1214 ;
0310  1215 ;       This routine receives control when a digit count larger than 31
0310  1216 ;       is detected. The exception is architecturally defined as an
0310  1217 ;       abort so there is no need to store intermediate state. All of the
0310  1218 ;       routines in this module save all registers R0 through R11 before
0310  1219 ;       performing the digit check. These registers must be restored
0310  1220 ;       before control is passed to VAX$ROPRAND.
0310  1221 ;
0310  1222 ; Input Parameters:
0310  1223 ;
0310  1224 ;       Entry at DECIMAL_ROPRAND
0310  1225 ;
0310  1226 ;               00(SP) - Return PC from common subroutine (discarded)
0310  1227 ;               04(SP) - Saved R0       \
0310  1228 ;                 .                      \
0310  1229 ;                 .                       > Restored
0310  1230 ;                 .                      /
0310  1231 ;               48(SP) - Saved R11      /
0310  1232 ;               52(SP) - Return PC from VAX$xxxxxx routine
0310  1233 ;
0310  1234 ;       Entry at DECIMAL_ROPRAND_NO_PC
0310  1235 ;
0310  1236 ;               00(SP) - Saved R0       \
0310  1237 ;                 .                      \
0310  1238 ;                 .                       > Restored
0310  1239 ;                 .                      /
0310  1240 ;               44(SP) - Saved R11      /
0310  1241 ;               48(SP) - Return PC from VAX$xxxxxx routine
0310  1242 ;
0310  1243 ; Output Parameters:
0310  1244 ;
0310  1245 ;       00(SP) - Offset in packed register array to delta PC byte
0310  1246 ;       04(SP) - Return PC from VAX$xxxxxx routine
0310  1247 ;
0310  1248 ;                The two flags in this longword (PACK_M_FPD and PACK_M_ACCVIO)
0310  1249 ;                are both clear in the case of a reserved operand abort.
0310  1250 ;
0310  1251 ; Implicit Output:
0310  1252 ;
0310  1253 ;       This routine passes control to VAX$ROPRAND where further
0310  1254 ;       exception processing takes place.
0310  1255 ;
0310  1256 ; Note:
0310  1257 ;
0310  1258 ;       This routine can be entered either from internal subroutines or from
0310  1259 ;       the callers of these subroutines. The DECIMAL_ROPRAND entry point is
0310  1260 ;       used when the return PC is on the stack because that is the name of
0310  1261 ;       the routine that is qutomatically invoked by the ROPRAND_CHECK macro
0310  1262 ;       when an illegal digit count is detected. The other name is arbitrary.
0310  1263 ;-
0310  1264 ;
0310  1265          ASSUME CVTPT_B_DELTA_PC EQ CVTPS_B_DELTA_PC
0310  1266          ASSUME CVTSP_B_DELTA_PC EQ CVTPS_B_DELTA_PC
0310  1267          ASSUME CVTTP_B_DELTA_PC EQ CVTPS_B_DELTA_PC
```

```
                                                M 1
                     0310  1268
                     0310  1269 DECIMAL_ROPRAND:
5E   04   C0         0310  1270                ADDL    #4,SP                   ; Discard return PC from common routine
                     0313  1271 DECIMAL_ROPRAND_NO_PC:
0FFF 8F   BA         0313  1272                POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
     03   DD         0317  1273                PUSHL   #CVTPS_B_DELTA_PC       ; Store offset to delta PC byte
  FCE4'   31         0319  1274                BRW     VAX$ROPRAND             ; Pass control along
```

```
                                031C  1276          .SUBTITLE       CONVERT_ACCVIO - Reflect an Access Violation
                                031C  1277 ;+
                                031C  1278 ; Functional Description:
                                031C  1279 ;
                                031C  1280 ;       This routine receives control when an access violation occurs while
                                031C  1281 ;       executing within the emulator routines for CVTPS, CVTPT, CVTSP, or
                                031C  1282 ;       CVTTP.
                                031C  1283 ;
                                031C  1284 ;       The routine header for ASHP_ACCVIO in module VAX$ASHP contains a
                                031C  1285 ;       detailed description of access violation handling for the decimal
                                031C  1286 ;       string instructions.
                                031C  1287 ;
                                031C  1288 ; Input Parameters:
                                031C  1289 ;
                                031C  1290 ;       See routine ASHP_ACCVIO in module VAX$ASHP
                                031C  1291 ;
                                031C  1292 ; Output Parameters:
                                031C  1293 ;
                                031C  1294 ;       See routine ASHP_ACCVIO in module VAX$ASHP
                                031C  1295 ;-
                                031C  1296
                                031C  1297 CONVERT_ACCVIO:
                          52    D4  031C  1298          CLRL    R2                      ; Initialize the counter
                      FCDE CF   9F  031E  1299          PUSHAB  MODULE_BASE             ; Store base address of this module
                        51  8E    C2  0322  1300          SUBL2   (SP)+,R1                ; Get PC relative to this base
                                0325  1301
                0000'CF42   51    B1  0325  1302 10$:     CMPW    R1,PC_TABLE_BASE[R2]    ; Is this the right PC?
                            07    13  032B  1303          BEQL    30$                     ; Exit loop if true
                         F4 52    29  F2  032D  1304          AOBLSS  #TABLE_SIZE,R2,10$      ; Do the entire table
                                0331  1305
                                0331  1306 ; If we drop through the dispatching based on PC, then the exception is not
                                0331  1307 ; one that we want to back up. We simply reflect the exception to the user.
                                0331  1308
                            0F  BA  0331  1309 20$:     POPR    #^M<R0,R1,R2,R3>        ; Restore saved registers
                            05  0333  1310          RSB                             ; Return to exception dispatcher
                                0334  1311
                                0334  1312 ; The exception PC matched one of the entries in our PC table. R2 contains
                                0334  1313 ; the index into both the PC table and the handler table. R1 has served
                                0334  1314 ; its purpose and can be used as a scratch register.
                                0334  1315
                51  0000'CF42  3C  0334  1316 30$:     MOVZWL  HANDLER_TABLE_BASE[R2],R1       ; Get the offset to the handler
                    FCC1 CF41   17  033A  1317          JMP     MODULE_BASE[R1]         ; Pass control to the handler
                                033F  1318
                                033F  1319 ; In all of the instruction-specific routines, the state of the stack
                                033F  1320 ; will be shown as it was when the exception occurred. All offsets will
                                033F  1321 ; be pictured relative to R0.
```

```
                                .SUBTITLE       Context-Specific Access Violation Handling
                        ;+
                        ; Functional Description:
                        ;
                        ;       It is relatively simple to back out any of these four instructions
                        ;       because their use of stack space is so simple. Each of the four
                        ;       routines contains a certain amount of initialization or completion
                        ;       code that uses no stack space (over and above the saved register
                        ;       array). Additional processing occurs one level deep in a subroutine
                        ;       where there is a return PC on the stack that must be discarded.
                        ;
                        ; Input Parameters:
                        ;
                        ;       R0 - Address of top of stack when access violation occurred
                        ;
                        ;       See specific entry points for details
                        ;
                        ; Output Parameters:
                        ;
                        ;       See input parameter list for VAX$DECIMAL_ACCVIO in module VAX$ASHP
                        ;-
                        ;
                        ;+
                        ; CVTPx_SAVED_R1
                        ;
                        ; An access violation occurred in routine CVTPx_COMMON along the code path
                        ; where the intermediate value of R1 is stored on the stack along with the
                        ; return PC. This must be disacrded.
                        ;
                        ;       00(R0) - Saved intermediate value of R1
                        ;       04(R0) - Return PC in mainline of VAX$CVTPS or VAX$CVTPT
                        ;       08(R0) - Saved R0
                        ;       12(R0) - Saved R1
                        ;        etc.
                        ;-
                        ;
                        CVTPx_SAVED_R1:
50   04   C0            ADDL    #4,R0                   ; Skip over saved R1 and drop into ...
                        ;
                        ;+
                        ; CONVERT_BSBW
                        ;
                        ; An access violation occurred somewhere in CVTPx_COMMON or CVTxP_COMMON.
                        ; The return PC must be discarded.
                        ;
                        ;       00(R0) - Return PC in VAX$CVTPS, VAX$CVTPT, VAX$CVTSP, or VAX$CVTTP
                        ;       04(R0) - Saved R0
                        ;       08(R0) - Saved R1
                        ;        etc.
                        ;-
                        ;
                        CVTPx_BSBW:
                        CVTxP_BSBW:
50   04   C0            ADDL    #4,R0                   ; Skip over return PC and drop into ...
                        ;
                        ;+
                        ; CONVERT_ACCVIO
```

```
                     0345  1380 ;
                     0345  1381 ; The access violation occurred in one of the four outer routines where
                     0345  1382 ; nothing other than the saved registers has been pushed onto the stack.
                     0345  1383 ; Nothing more needs to be done to the registers or the stack before
                     0345  1384 ; transferring control to VAX$DECIMAL_ACCVIO. These entry points are merely
                     0345  1385 ; a convenience.
                     0345  1386 ;
                     0345  1387 ;       00(SP) - Saved R0
                     0345  1388 ;       04(SP) - Saved R1
                     0345  1389 ;       08(SP) - Saved R2
                     0345  1390 ;       12(SP) - Saved R3
                     0345  1391 ;        etc.
                     0345  1392 ;-
                     0345  1393
                     0345  1394 CVTPS_ACCVIO:
                     0345  1395 CVTPT_ACCVIO:
                     0345  1396 CVTSP_ACCVIO:
                     0345  1397 CVTTP_ACCVIO:
FCB8' 31             0345  1398         BRW     VAX$DECIMAL_ACCVIO      ; Join common code to restore registers
                     0348  1399
                     0348  1400         .END
```

```
...PC...                        = 00000306
...ROPRAND...                   = 0000027C R      02
CONVERT_ACCVIO                    0000031C R      02
CVTPS_ACCVIO                      00000345 R      02
CVTPS_A_DSTADDR                 = 0000000C
CVTPS_B_DELTA_PC                = 00000003
CVTPT_ACCVIO                      00000345 R      02
CVTPT_B_DELTA_PC                = 00000003
CVTPT_V_FPD                     = 0000000F
CVTPX_BSBW                        00000342 R      02
CVTPX_COMMON                      000000B6 R      02
CVTPX_EQUAL                       00000125 R      02
CVTPX_OVERFLOW_CHECK              000000D8 R      02
CVTPX_SAVED_R1                    0000033F R      02
CVTPX_TABLE                       00000000 R      02
CVTPX_ZERO_FILL                   0000011C R      02
CVTSP_ACCVIO                      00000345 R      02
CVTSP_A_SRCADDR                 = 00000004
CVTSP_B_DELTA_PC                = 00000003
CVTTP_ACCVIO                      00000345 R      02
CVTTP_B_DELTA_PC                = 00000003
CVTTP_V_FPD                     = 0000000F
CVTXP_BSBW                        00000342 R      02
CVTXP_COMMON                      00000277 R      02
CVTXP_EQUAL                       000002D4 R      02
CVTXP_OVERFLOW_CHECK              0000029E R      02
CVTXP_TABLE_HIGH                  0000017D R      02
CVTXP_TABLE_LOW                   00000173 R      02
CVTXP_ZERO_FILL                   000002B9 R      02
DECIMAL_ROPRAND                   00000310 R      02
DECIMAL_ROPRAND_NO_PC             00000313 R      02
HANDLER_TABLE_BASE                00000000 R      04
MODULE_BASE                     = 00000000 R      02
PC_TABLE_BASE                     00000000 R      03
PSL$M_N                         = 00000008
PSL$M_V                         = 00000002
PSL$M_Z                         = 00000004
PSL$V_V                         = 00000001
PSL$V_Z                         = 00000002
TABLE_SIZE                      = 00000029
VAX$CVTPS                         00000010 RG     02
VAX$CVTPT                         0000006F RG     02
VAX$CVTPT_JSB                     00000064 RG     02
VAX$CVTPT_RESTART                 00000068 RG     02
VAX$CVTSP                         00000187 RG     02
VAX$CVTTP                         00000212 RG     02
VAX$CVTTP_JSB                     00000207 RG     02
VAX$CVTTP_RESTART                 0000020B RG     02
VAX$DECIMAL_ACCVIO                ********    X   00
VAX$DECIMAL_EXIT                  ********    X   00
VAX$ROPRAND                       ********    X   00
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
VAX$CODE	00000348 (840.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG
PC_TABLE	00000052 (82.)	03 (3.)	PIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
HANDLER_TABLE	00000052 (82.)	04 (4.)	PIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	15	00:00:00.07	00:00:01.87
Command processing	77	00:00:00.50	00:00:04.79
Pass 1	166	00:00:05.44	00:00:19.35
Symbol table sort	0	00:00:00.16	00:00:00.82
Pass 2	248	00:00:02.79	00:00:12.15
Symbol table output	7	00:00:00.06	00:00:00.06
Psect synopsis output	2	00:00:00.03	00:00:00.45
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	515	00:00:09.05	00:00:39.49

The working set limit was 1200 pages.
31903 bytes (63 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 127 non-local and 60 local symbols.
1400 source lines were read in Pass 1, producing 21 object records in Pass 2.
19 pages of virtual memory were used to define 17 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[EMULAT.OBJ]VAXMACROS.MLB;1	9
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	14

249 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:VAXCONVRT/OBJ=OBJ$:VAXCONVRT MSRC$:VAXCONVRT/UPDATE=(ENH$:VAXCONVRT)+LIB$:VAXMACROS/LIB
```

FPLOAD
LIS

VAXASHP
LIS

VAXCONVRT
LIS

VAXARITH
LIS

VAXHANDLR
LIS

VAXCVTLP
LIS

VAXDECIML
LIS

VAXEMULAT
LIS

VAXCVTPL
LIS

VAXEDITPC
LIS